REMARKS

The present amendment and remarks are in response to the Non-Final Office Action entered in the above-identified case and mailed on January 25, 2010. Claims 1-37 are pending in the application. Claims 1, 8-15, 20-26 and 32-37 were rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 7,086,009 to Resnick, et al. (hereafter "Resnick") in view of U.S. Patent No. 5,926,177 to Hatanaka et al. (hereafter "Hatanaka"). Claims 2-7, 16-19 and 27-31 were rejected under 35 U.S.C. §103(a) as being unpatentable over Resnick in view of Hanataka and further in view of U.S. Patent No. 5,485,600 to Joseph et al. (hereafter "Joseph"). With respect to each ground for rejection Applicants respectfully traverse.

Applicants turn first to the rejection of independent claim 1. A claim is unpatentable under 35 U.S.C. §103(a) only if every element of the claims is found in the prior art. In the present case, claim 1 is not unpatentable over the combined teaching of Resnick and Hanataka because even when combined, Resnick and Hanataka do not teach each and every element of claim 1.

Claim 1 calls for, among other things, a definition routine adapted to enable a user to define a routine that operates in conjunction with the graphic representation of a process entity object and a value of a property associated with a process entity during execution of the display object, as well as a routine that operates in conjunction with the graphic representation of the process entity to alter the manner in which the graphic representation of the process entity is displayed to the user according to the value of the property to reflect an operating condition of the process plant associated with the process entity. These features are not taught or suggested by Resnick or Hatanaka either standing alone or in combination.

The Examiner points to Resnick col. 29, line 61-col. 30 line 26 as teaching a definition routine adapted to enable a user to define a routine that operates in conjunction with a visual representation of a graphic object and a property associated with a process entity during execution of the graphic display. A close inspection of this passage, however, shows that Resnick does not in fact disclose these features of the claimed invention. Resnick col. 29, line 61-col. 30, line 26 describes a software development toolkit. The toolkit includes interface component definitions, components, and utilities. The interface definitions include a configuration interface, a runtime interface, and a package interface. The interface definitions allow a single object to provide interaction with, for example, an editing environment, a validation environment, a runtime environment, or other environments. The configuration interface supports methods that allow an object's configuration to be edited in an Integrated Development Environmnet (IDE). The package interface supports methods that enable an object to be packaged such that the packaged object can be imported, exported, uploaded downloaded, etc. The passage cited by the Examiner further describes object attributes that are accessible by message exchange. Object attributes may be classified as configuration or state attributes. The Runtime Interface includes methods that may be executed on identified attributes of an object. The methods provide behaviors to the objects during runtime. The Runtime Interface includes, for example a GetAttribute method and a SetAttribute method. A GetAttribute call returns requested data from an object, while a SetAttribute method sets a value or invokes an action by an object. The SetAttribute call also optionally returns a response value. Resnick col. 29, line 61-col. 30, line 25 is set forth in its entirety below.

> Turning now to FIG. 20, a set of blocks depict the functional components of a toolkit 2000. The toolkit 2000 includes interface component definitions 2002, components 2004, and utilities 2006. The interface

> definitions 2002 include a Configuration (editor) Interface 2010, a Runtime Interface 2012, and Package Interface 2014. The set of interface definitions 2002 enable a single object to provide useful interaction with a variety of environments (e.g., editing, validation, runtime, etc.). The Configuration Interface 2010 supports a set of methods enabling an object's configuration to be edited by the IDE. The Package Interface 2014 supports a set of methods enabling the object to be packaged and then imported, exported, uploaded, downloaded, copied, and/or stored within the configuration database 124.
>
> Attributes are the portions of an object that are accessible by message exchange. They generally are classified as configuration (set at the time of configuration) and state (set during runtime). The Runtime Interface 2012 supports a set of methods executed on identified attributes that provide behaviors to the object during runtime when an object is deployed on a host application engine. The runtime interface 2012 comprises a GetAttribute and a SetAttribute method enabling other objects to access exposed attributes on the object. A GetAttribute call includes the following inputs: attribute handle, access level (for security), and locale ID (for localizing language). The GetAttribute call returns the requested data and quality of the data. The SetAttribute method is used to set a value and/or invoke an action by the object (a pseudo-remote procedure call. The SetAttribute call includes the following inputs: attribute handle, confirmation handle, write type, access level, access level verifier, data in. The SetAttribute call returns an optional response value.

Nowhere does the cited passage describe a routine that enables a user to define another routine that operates in conjunction with a graphic representation of a process entity and the value of a property associated with the process entity.

Next, the Examiner points to Resnick col. 30, line 50 – col. 31, line 16 as teaching a routine that operates in conjunction with a graphic representation of a process entity displayed to a user and that is associated with the value of a property. Again, however, the cited passage does not actually teach the features described by the Examiner. The cited passage actually teaches utilities, including an Object Designer, that support an object template development process. Using the Object Designer a developer can specify primitives and attributes within an object. In addition to allowing the developer to specify attribute values, the Object Designer also allows new attributes to be defined. According to an

embodiment, the Object Designer supports a basic mode and an advanced mode. The basic mode facilitates the creation of application objects that include common primitives, custom primitives and utility primitives. The advanced mode facilitates the creation of new primitives, a hierarchy of primitives, or an object comprising a primitive hierarchy. The cited passage further describes a set of code wizards. Code wizards generate source code for a specified target language based upon an object or primitive definition rendered by the Object Designer. This results in wrapper classes for the object. The wrapper classes for the object expose the object as being defined in the target language. The full text of Resnick col. 30, line 50 – col. 31, line 16 is printed below.

> The utilities 2006 support the object template development process. An Object Designer 2020 is a graphical utility that enables a developer to develop new objects and primitives for objects. A developer, through the Object Designer 2020, specifies primitives and attributes within an object or primitive. The Object Designer 2020 facilitates configuring an object template's attributes, specifying supporting primitives (including the utility primitives identified herein above), and configuring the attributes of the associated primitives. The Object Designer 2020, in addition to allowing attribute values to be specified, allows totally new attributes to be defined.
>
> In an embodiment of the invention, the Object Designer 2020 supports a basic mode and advanced mode. The basic mode facilitates creating application objects comprising the common primitive, a custom primitive, and utility primitives. The advanced mode facilitates creating new primitives, or a hierarchy of primitives, and/or an object comprising a hierarchy of primitives. The object designer also enables a user to launch a code wizard that generates skeleton code for a primitive.
>
> A set of code wizards 2024 generate source code for a specified target language based upon an object or primitive definition rendered by the Object Designer 2020. Resulting wrapper classes for the object will expose the object as being defined in the target language. In the case of C++ code, an attribute class is provided and a custom object class containing attributes are generated. The attributed class contains assignment and casting operators that allow appropriate rows within the attribute table to be accessed when the wrapper class's attributes are accessed. In the case of Visual Basic code, an attribute class is provided and a customer object class containing attributes is generated. The attribute class contains Get and Set Property routines that access the proper row of an attribute table.

This passage simply does not disclose a routine that operates in conjunction with the graphic representation of a process entity displayed to a user and that is associated with the value of a property of the process entity. The Object Designer facilitates creating application objects, it does not operate in conjunction with graphic representations of process entities. The closest thing to a routine described in the cited passages is probably the code wizards. The code wizards, however, simply generate source code for a target language based on objects or primitives that have been rendered by object designer. Again, the code wizards do not operate in conjunction with the graphic representations of process entities displayed to a user and that are associated with the value of a properties of the process entities when a graphic display is executed.

Hatanaka discloses a system for providing multiple views on a display device in a model-view-controller architecture. Hatanaka is cited for teaching altering the manner in which a graphic representation of a process entity is displayed to a user. Specifically, the Examiner points to Hatanaka col. 3, line 58-col. 4, line 45 as teaching this feature. There Hatanaka describes a ViewProxy that implements a standard view interface in a model-view-controller architecture. The ViewProxy manages a collection of view objects. The ViewProxy forwards all notifications it receives from a model and controller to each of the views it manages. When the state of an object changes, the model notifies the view which, in a single view environment, would cause the display to change. In the multiple view environment, the ViewProxy notifies all of the views that display the object so that each view can change its display model to reflect the changed state. The user then sees the change in the active display. Thus, Hatanaka teaches a display that alters the manner in which a graphic representation of a model is displayed, but the graphic representation is not a graphic

representation of a process entity, and the change in the manner in which the graphic representation of the process entity is displayed to the user is not based on the value of a property associated with a process entity to reflect an operating condition of a process plant wherein the reflected operating condition is associated with the process entity as called for in claim 1.

As the preceding discussion shows, the combined disclosure of Resnick and Hatanaka does not teach each and every element of claim 1 of the present application. Accordingly, claim 1 is not unpatentable under 35 U.S.C. §103(a) over Resnick and Hatanaka and should be allowed. Furthermore, since independent claims 15 and 16 include features similar to those described above with regard to claim 1, and because these claims where rejected on similar grounds as claim 1, independent claims 15 and 26 are also not unpatentable under 35 U.S.C. §103(a) for the same reasons. Claims 8-14, 20-25, and 32-37 all depend either directly or indirectly from claims 1, 15, and 26, and are also allowable for the same reasons as claims 1, 15, and 26.

Next Applicants turn to the rejection of claims 2-7, 16-19, and 27-31 under 35 U.S.C. §103(a) as being unpatentable over Resnick in view of Hatanaka and further in view of Joseph. As already discussed, Resnick and Hatanaka do not disclose every element of independent claims 1, 15, and 26. Joseph is cited as teaching various features of the dependent claims. For example, with regard to claims 3, 17, and 28 Joseph is cited as teaching an animation routine that animates a graphic representation in a continuous manner. With regard to claims 4, 18, and 19 Joseph is cited as teaching an animation routine that applies skew, rotation and resizing to a graphic representation. With regard to claims 5, 19, and 30 Joseph is cited as teaching a color animation, a color gradient animation, opacity

animation, or a font characteristic, or a video property of the graphic representation. Finally, with regard to claims 6 and 31 Joseph is cited as teaching graphic representations that include two or more primitives and a routine that changes a property of one of the primitives.

Applicants argue that Joseph does not teach the features of the dependent claims identified by the Examiner. However, even assuming *arguendo* that Joseph does teach these features, neither Resnick, Hatanaka nor Joseph teach a definition routine adapted to enable a user to define a routine that operates in conjunction with a graphical representation of one of the graphic objects and property associated with a process entity. Since neither Resnick, Hatanaka nor Joseph teach this feature of the independent claims, it follows that the combined teaching of Resnick, Hatanaka and Joseph does not teach or suggest all of the features of any of the dependant claims. Accordingly claims 2-7, 16-19 and 27-31 are not obvious over Resnick, Hatanaka and Joseph and should be allowed.

CONCLUSION

For the foregoing reasons, Applicants respectfully submit that all of the claims pending in the application are now in condition for allowance. If the Examiner has any questions regarding the present response he is encouraged to call the Applicant's attorney at the number provided below.

Dated: March 9, 2010

Respectfully submitted,

By: /Jeffrey H. Canfield #38,404/
Jeffrey H. Canfield
Registration No.: 38,404
MARSHALL, GERSTEIN & BORUN LLP
233 S. Wacker Drive, Suite 6300
Sears Tower
Chicago, Illinois 60606-6357
(312) 474-6300
Attorney for Applicant